UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/847,067	04/30/2001	Brian T. Murren	GE1-005US	4549

21718 7590 05/01/2008

LEE & HAYES PLLC
SUITE 500
421 W RIVERSIDE
SPOKANE, WA 99201

EXAMINER	
DESAI, RACHNA SINGH	
ART UNIT	**PAPER NUMBER**
2176	

NOTIFICATION DATE	DELIVERY MODE
05/01/2008	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

lhpto@leehayes.com

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Application Number: 09/847,067
Filing Date: April 30, 2001
Appellant(s): MURREN ET AL.

David S. Lee
For Appellant

EXAMINER'S ANSWER

This is in response to the appeal brief filed 02/14/08 appealing from the Office action mailed 02/21/07.

(1) Real Party in Interest

A statement identifying by name the real party in interest is contained in the brief.

(2) Related Appeals and Interferences

The examiner is not aware of any related appeals, interferences, or judicial proceedings which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

(3) Status of Claims

The statement of the status of claims contained in the brief is correct.

(4) Status of Amendments After Final

The appellant's statement of the status of amendments after final rejection contained in the brief is correct.

(5) Summary of Claimed Subject Matter

The summary of claimed subject matter contained in the brief is correct.

(6) Grounds of Rejection to be Reviewed on Appeal

The appellant's statement of the grounds of rejection to be reviewed on appeal is correct.

(7) Claims Appendix

The copy of the appealed claims contained in the Appendix to the brief is correct.

(8) Evidence Relied Upon

2004/0039993 A1	KOUGIOURIS et al.	02-2004

(9) Grounds of Rejection

The following ground(s) of rejection are applicable to the appealed claims:

Claim Rejections - 35 USC § 112

Claims 37-38 rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the written description requirement. The claim(s) contains subject matter which was not described in the specification in such a way as to reasonably convey to one skilled in the relevant art that the inventor(s), at the time the application was filed, had possession of the claimed invention.

Examiner is unable to find support for the limitations of claims 37-38. For example, the Specification does not appear to discuss the form generation procedure ***being independent or antecedent to a user's interaction*** with the computer program.

Claim Rejections - 35 USC § 102

The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

Claims 1-2, 4-15, 17-19, 30-34 and 37-38 are rejected under 35 U.S.C. 102(e) as being anticipated by Kougiouris et al., US 2004/0039993 A1, 2/26/04 (Filed 8/27/03, Continuation of application filed 11/15/99).

In reference to claim 1, Kougiouris teaches an automatic formatting and validating of text for markup language graphical user interface (GUI). The GUI markup language description comprises various types of GUI elements for which text is to be validated and formatted such as form fields, tables, and links. See page 1, paragraph [0010]. The GUI element may comprise one or more fields for accepting text input and displaying text output. The markup language file GUI descriptions comprise information usable by the validation/formatting manager component to perform various types of validating/formatting operations. This information may exist as markup language tag attributes, e.g., by adding custom attributes to markup language. See page 4,

paragraphs [0061]-[0064]. The user may provide text input to the GUI element which is validated by the manager before it is displayed in HTML form. See page 5, paragraph [0070]-[0075]. This meets the limitations, ***accessing a computer program; automatically identifying a set of one or more attributes of the computer program with values that are to be input to the computer program by a user***. Kougiouris further discloses that Graphical user interfaces (GUIs) often include text fields for accepting text input or displaying text output. For example, graphical user interfaces may comprise a "form", that is a series of text fields with a look and feel similar to a paper-based form. Many text fields are designed to accept text input or display text output that is often formatted or demarcated in a particular way. See page 1, paragraphs [0005]. The user may provide text input to the GUI element which is validated by the manager before it is displayed in HTML form. See page 5, paragraph [0070]-[0075]. The graphical user interfaces can be created from markup languages such as HTML or XML-derived markup language descriptions which meets the limitation, ***creating code for one or more forms including selected ones of the set of one or more attributes***.

In reference to claim 2, Kougiouris discloses outputting the attributes in a form such as an HTML form in which the various attributes are listed. See figures 5A-5C.

In reference to claims 4-5, Kougiouris discloses that Graphical user interfaces (GUIs) often include text fields for accepting text input or displaying text output. For

example, graphical user interfaces may comprise a "form", that is a series of text fields with a look and feel similar to a paper-based form. Many text fields are designed to accept text input or display text output that is often formatted or demarcated in a particular way. See page 1, paragraphs [0005]. The user may provide text input to the GUI element which is validated by the manager before it is displayed in HTML form. See page 5, paragraph [0070]-[0075]. The graphical user interfaces can be created from markup languages such as HTML or XML-derived markup language descriptions.

In reference to claim 6, Kougiouris discloses the user may provide text input to the GUI element which is validated by the manager before it is displayed in HTML form. See page 5, paragraph [0070]-[0075]. See also figures 5A-5C which illustrate a data input field for inputting a value for the attributes. The user may also perform various other actions causing the application to check the text, such as issuing a command to submit the data the user has entered to a database, or perform other types of transactions using the data. See page 8, paragraph [0125].

In reference to claim 7, Kougiouris teaches the user may perform various other actions causing the application to check the text, such as issuing a command to submit the data the user has entered to a database, or perform other types of transactions using the data. See page 8, paragraph [0125].

In reference to claim 8, Kougiouris discloses the information may exist as markup language tag attributes, e.g., by adding custom attributes to markup language. See page 4, paragraphs [0061]-[0064]. The user may provide text input to the GUI element which is validated by the manager before it is displayed in HTML form. See page 5, paragraph [0070]-[0075].

In reference to claim 9, Kougiouris discloses outputting the attributes in a form such as an HTML form in which the various attributes are listed. See figures 5A-5C.

In reference to claim 10, Kougiouris discloses that Graphical user interfaces (GUIs) often include text fields for accepting text input or displaying text output. For example, graphical user interfaces may comprise a "form", that is a series of text fields with a look and feel similar to a paper-based form. Many text fields are designed to accept text input or display text output that is often formatted or demarcated in a particular way. See page 1, paragraphs [0005]. The user may provide text input to the GUI element which is validated by the manager before it is displayed in HTML form. See page 5, paragraph [0070]-[0075]. The graphical user interfaces can be created from markup languages such as HTML or XML-derived markup language descriptions.

In reference to claim 11, Kougiouris teaches an automatic formatting and validating of text for markup language graphical user interface (GUI). The GUI markup language description comprises various types of GUI elements for which text is to be

validated and formatted such as form fields, tables, and links. See page 1, paragraph [0010]. The GUI element may comprise one or more fields for accepting text input and displaying text output. The markup language file GUI descriptions comprise information usable by the validation/formatting manager component to perform various types of validating/formatting operations. This information may exist as markup language tag attributes, e.g., by adding custom attributes to markup language. See page 4, paragraphs [0061]-[0064]. The user may provide text input to the GUI element which is validated by the manager before it is displayed in HTML form. See page 5, paragraph [0070]-[0075]. Kougiouris teaches the user may perform various other actions causing the application to check the text, such as issuing a command to submit the data the user has entered to a database, or perform other types of transactions using the data. See page 8, paragraph [0125]. This teachings meets the limitations, ***accessing a computer program, wherein the computer program includes a plurality of interactions that each include one or more command definitions and one or more view definitions, wherein each command definition defines a command having various attributes and a behavior, and wherein each view definition defines a view that is a response to a request; automatically identifying a set of one or more attributes of the computer program with values that are to be input to the computer program by a user***.

Kougiouris teaches the user may perform various other actions causing the application to check the text, such as issuing a command to submit the data the user has entered to a database, or perform other types of transactions using the data. See

page 8, paragraph [0125]. This meets the limitation, ***identifying, for each of the command definitions of each of a plurality of interactions, the methods of the command definition.*** Kougiouris discloses the user may provide text input to the GUI element which is validated by the manager before it is displayed in HTML form. See page 5, paragraph [0070]-[0075]. See also figures 5A-5C which illustrate a data input field for inputting a value for the attributes. The user may also perform various other actions causing the application to check the text, such as issuing a command to submit the data the user has entered to a database, or perform other types of transactions using the data. See page 8, paragraph [0125].

Kougiouris teaches that when a user performs an action causing the application to check the text entered by the user, the application **invokes component method in order to validate the text the user entered**. If the text is valid, the application **invokes a component method in order to format the text the user has entered** which meets the limitation, ***checking, for each identified method that sets a value, whether a corresponding identified method obtains the value, and identifying, as an attribute of the set of one or more attributes, each attribute corresponding to a method that sets a value for the attribute for which there is no corresponding identified method that obtains the value for the attribute***. See figure 7. Kougiouris discloses outputting the attributes in a form such as an HTML form in which the various attributes are listed which meets the limitation, ***outputting an identification of the set of one or more attributes***. See figures 5A-5C.

In reference to claim 12, Kougiouris teaches that when a user performs an action causing the application to check the text entered by the user, the application **invokes component method in order to validate the text the user entered**. If the text is valid, the application **invokes a component method in order to format the text the user has entered** which meets the limitation, ***wherein identifying the methods of the command definition comprises querying the command definition to cause the command definition to identify its own methods***. See figure 7.

In reference to claim 13, Kougiouris discloses GUI elements comprising user-interface elements where the attributes are default attributes. See page 5, paragraph [0067].

In reference to claim 14, Kougiouris teaches an automatic formatting and validating of text for markup language graphical user interface (GUI). The GUI markup language description comprises various types of GUI elements for which text is to be validated and formatted such as form fields, tables, and links. See page 1, paragraph [0010]. The GUI element may comprise one or more fields for accepting text input and displaying text output. The markup language file GUI descriptions comprise information usable by the validation/formatting manager component to perform various types of validating/formatting operations. This information may exist as markup language tag attributes, e.g., by adding custom attributes to markup language. See page 4, paragraphs [0061]-[0064]. The user may provide text input to the GUI element which is

validated by the manager before it is displayed in HTML form. See page 5, paragraph [0070]-[0075]. The GUI is generated using HTML or XML code as shown in figures 5A-5C. This meets the limitations, ***accessing a computer program; automatically identifying a set of one or more outputs of the computer program.*** Kougiouris discloses outputting the attributes in a form such as an HTML form in which the various attributes are listed. See figures 5A-5C. This meets the limitation ***generating a list identifying the set of one or more outputs; and outputting the list***. The said identifying one or more outputs of a computer program is ***an analysis of the computer code***.

In reference to claim 15, Kougiouris teaches an automatic formatting and validating of text for markup language graphical user interface (GUI). The GUI markup language description comprises various types of GUI elements for which text is to be validated and formatted such as form fields, tables, and links. See page 1, paragraph [0010]. The GUI element may comprise one or more fields for accepting text input and displaying text output. The markup language file GUI descriptions comprise information usable by the validation/formatting manager component to perform various types of validating/formatting operations. This information may exist as markup language tag attributes, e.g., by adding custom attributes to markup language. See page 4, paragraphs [0061]-[0064]. The user may provide text input to the GUI element which is validated by the manager before it is displayed in HTML form. See page 5, paragraph [0070]-[0075]. This meets the limitation, ***automatically identifying a set of one or***

more attributes of the computer program with values that are to be input to the computer program by a user; and outputting an identification of the set of one or more attributes. Kougiouris further discloses that Graphical user interfaces (GUIs) often include text fields for accepting text input or displaying text output. For example, graphical user interfaces may comprise a "form", that is a series of text fields with a look and feel similar to a paper-based form. Many text fields are designed to accept text input or display text output that is often formatted or demarcated in a particular way. See page 1, paragraphs [0005]. The user may provide text input to the GUI element which is validated by the manager before it is displayed in HTML form. See page 5, paragraph [0070]-[0075]. The graphical user interfaces can be created from markup languages such as HTML or XML-derived markup language descriptions.

In reference to claim 17, Kougiouris further discloses that Graphical user interfaces (GUIs) often include text fields for accepting text input or displaying text output. For example, graphical user interfaces may comprise a "form", that is a series of text fields with a look and feel similar to a paper-based form. Many text fields are designed to accept text input or display text output that is often formatted or demarcated in a particular way. See page 1, paragraphs [0005]. The user may provide text input to the GUI element which is validated by the manager before it is displayed in HTML form. See page 5, paragraph [0070]-[0075]. The graphical user interfaces can be created from markup languages such as HTML or XML-derived markup language descriptions

which meets the limitation, ***creating one or more forms included selected ones of the set of one or more outputs.***

In reference to claims 18-19, Kougiouris teaches the user may perform various other actions causing the application to check the text, such as issuing a command to submit the data the user has entered to a database, or perform other types of transactions using the data. See page 8, paragraph [0125].

Regarding claim 30, Kougiouris teaches generating a GUI created using XML or HTML markup language code where the GUI displays attribute values and allows the values to be set by a user which meets the limitation, ***accessing the computer program to identify operations in the computer program that load attribute values and set attribute values***. See page 1, paragraph [0010] and figures 5a-5c. The GUI markup language description comprises various types of GUI elements for which text is to be validated and formatted such as form fields, tables, and links. See page 1, paragraph [0010]. The GUI element may comprise one or more fields for accepting text input and displaying text output which meets the limitation, ***analyzing the identified operations to determine one or more user inputs to the computer program; automatically generating code for one or more input forms to allow a user to input at least some of the one or more user inputs***. See figures 5A-5C. The user may provide text input to the GUI element which is validated by the manager before it is displayed in HTML form which meets the limitation, ***accessing the computer program***

to identify one or more outputs of the computer program; automatically generating code for one or more output forms to present the outputs of the computer program. See page 5, paragraph [0070]-[0075]. Kougiouris further discloses that Graphical user interfaces (GUIs) often include text fields for accepting text input or displaying text output. For example, graphical user interfaces may comprise a "form", that is a series of text fields with a look and feel similar to a paper-based form. The user may provide text input to the GUI element which is validated by the manager before it is displayed in HTML form. See page 5, paragraph [0070]-[0075].

Claims 31-34 are rejected under the same rationale used in claims 2, 6, 11, and 13 respectively above.

Regarding claims 37-38, Kougiouris teaches presenting a GUI with a form including text fields for accepting text input. The GUI may comprise a form that is a series of text fields with a look and feel similar to a paper-based form. See page 1, paragraph [0005].

(10) Response to Argument

First Ground of Rejection

On pages 10-11 of the Brief, Appellant argues there is support for claims 37-38 on page 30, lines 4-12, page 34, lines 8-18, and page 36, lines 21-24.

Examiner disagrees.

Page 30 discusses automatically generating forms from the business logic. Page 34 discusses the test module and how it flags various conditions it detects from analyzing methods of the interaction. Finally, page 36 discusses the test module provides results of its analysis to a form creation module to create forms. While all three sections discuss automatically generating forms, there isn't necessarily any teaching that the form generation is independent and antecedent to a user's interaction with a program. After all, everything "automatic" must be initiated at some point in time with user interaction. Furthermore, independent claim 1 requires accessing a program and **user input** of attributes from which the code for forms is created. Therefore, it is unclear how the form generation process is completely independent and antecedent to user's interaction with the forms. In view of the comments above, the rejections under 35 U.S.C. 112, first paragraph are maintained.

Second Ground of Rejection

On pages 12-20, Appellant argues the rejections under 35 U.S.C. 102(e).

i. Kougiouris discloses a method for validating and formatting procedures for a GUI described in a markup language.

On pages 12-13, Appellant provides their interpretation of the Kougiouris reference.

ii. Kougiouris does not disclose, either directly or inherently "creating code" as contended by the Examiner.

On pages 14-16, Appellant begins arguments stating Kougiouris does not teach "creating code" and instead teaches "instantiating" an already encoded markup file (page 14). Appellant further argues Kougiouris merely takes already encoded HTML elements and displays the element as a text box in a GUI. Appellant states Kougiouris merely represents elements according to the markup file and the code which causes the computer to display a GUI is already in existence and therefore is not created but merely displayed (page 15). Finally, Appellant states in Kougiouris the coding already exists as HTML or XML elements and is not created **when** the markup file is displayed as a GUI. Appellant argues in order for the rejection to stand, the markup language file

cannot be encoded as this would not be a "creation at the point of the computer displaying the GUI".

Examiner disagrees with Appellant's statements.

As an initial point, the **timing** of the creation of code is not claimed nor required by the claims. All that is required is that code for one or more forms is created. It does not matter at what point in time this creation occurs. In Kougiouris, certainly the HTML/XML code used to display the GUI must be created at some point in time. Appellant even states on page 15, Kougiouris represents elements according to the markup file and **the code which causes the computer to display a GUI is already in existence** and therefore is not created but merely displayed (page 15). The claim does not specify when the code must be created and in order for the code to exist, it must be created.

The graphical user interfaces can be created from markup languages such as HTML or XML-derived markup language descriptions and code which meets the limitation, ***creating code for one or more forms including selected ones of the set of one or more attributes***. The markup language code used to create the GUI are created at some point in time. Therefore, Kougiouris does teach creating code.

iii. Claims 2, 4-10, and 37 are allowable as depending from an allowable base claim.

Appellant states each of the above claims depending from claim 1 are allowable as depending from an allowable base claim. The rejections of these claims are maintained in light of the comments above in section **ii** with respect to claim 1.

iv. Kougiouris discloses a method for validating and formatting user entered text. Kougiouris does not disclose whether a corresponding identified method obtains the value.

On pages 16-17, Appellant argues Kougiouris validates user input but does not check each identified method. Appellant states that Kougiouris' checking is not for an identified method but for user input. Appellant says that "while the claim language is plain on its face, the detailed description discloses this type of technique at page 32, line 25 through page 33, line 25".

Examiner disagrees.

The claim recites ***"checking, for an identified method that sets a value, whether a corresponding identified method obtains the value"***. Kougiouris teaches that when a user performs an action causing the application to check the text entered by the user, the application **invokes component method in order to validate**

the text the user entered. If the text is valid, the application **invokes a component method in order to format the text the user has entered**. See figure 7. Kougiouris teaches checking for an identified method that obtains the value in that the component method that formats the text obtains the value from the user inputted value. Therefore, Kougiouris teaches checking whether a corresponding method obtains a value.

v. Claims 12 and 13 depend either directly or indirectly from claim 11 and are allowable as depending from an allowable base claim, as well as their own recited features.

Appellant states each of the above claims depending from claim 13 are allowable as depending from an allowable base claim. The rejections of these claims are maintained in light of the comments above in section **iv** with respect to claim 13.

vi. Kougiouris does not disclose, either directly or inherently, "identifying and generating are performed based on analysis of computer program code, independent of execution of the computer program to provide one or more views" as contended by the Examiner.

On pages 18-19 of the Brief, Appellant argues Kougiouris does not disclose ***identifying and generating are performed based on an analysis of the computer program code independent of execution of the program to provide one or more views.*** Appellant argues Kougiouris only discloses analyzing user input which requires

the GUI be executing as text is analyzed. Appellant further argues Kougiouris simply does not analyze computer code.

Examiner disagrees.

Kougiouris discloses outputting the attributes in a form such as an HTML form in which the various attributes are listed. See figures 5A-5C. This meets the limitation ***generating a list identifying the set of one or more outputs; and outputting the list***. The said identifying one or more outputs of a computer program is ***an analysis of the computer code***. Since, Kougiouris discloses outputting the attributes in a form such as an HTML form in which the various attributes are listed (figures 5a-5c), the computer code is analyzed in order to determine what outputs would be generated and displayed within the GUI. The computer program is the markup language code used to generate the GUI as previously stated with respect to claim 1. The user input has nothing to do with the list identifying one or more outputs of a computer program. The GUI is generated from a computer program (i.e. XML/HTML code) which is also able to identify the one or more outputs. See figures 5A-5C of Kougiouris.

vii. Claims 15 and 17-19 depend either directly or indirectly from claim 14 and are allowable as depending from an allowable base claim, as well as for its own recited features.

Appellant states each of the above claims depending from claim 14 are allowable as depending from an allowable base claim. The rejections of these claims are maintained in light of the comments above in section **vi** with respect to claim 14.

ix. The Final action does not address each and every limitation of the claims and fails to make a prima facie case of anticipation. In particular the Final Action fails to address the linguistic differences appearing in independent claim 30 in comparison with claim 1.

Appellant argues Kougiouris fails to teach automatically generating code or analyzing the identified operations as claimed in claim 30.

Examiner disagrees.

Kougiouris teaches generating a GUI created using XML or HTML markup language code where the GUI displays attribute values and allows the values to be set by a user. See page 1, paragraph [0010] and figures 5a-5c. The GUI markup language description comprises various types of GUI elements for which text is to be validated and formatted such as form fields, tables, and links. See page 1, paragraph [0010]. The GUI element may comprise one or more fields for accepting text input and

displaying text output which meets the limitation, ***analyzing the identified operations to determine one or more user inputs to the computer program; automatically generating code for one or more input forms to allow a user to input at least some of the one or more user inputs***. See figures 5A-5C. The user may provide text input to the GUI element which is validated by the manager before it is displayed in HTML form which meets the limitation, ***accessing the computer program to identify one or more outputs of the computer program; automatically generating code for one or more output forms to present the outputs of the computer program***. See page 5, paragraph [0070]-[0075]. Kougiouris further discloses that Graphical user interfaces (GUIs) often include text fields for accepting text input or displaying text output. For example, graphical user interfaces may comprise a "form", that is a series of text fields with a look and feel similar to a paper-based form. The user may provide text input to the GUI element which is validated by the manager before it is displayed in HTML form. See page 5, paragraph [0070]-[0075].

Therefore, Kougiouris teaches generating XML/HTML code to present a GUI with input fields based on the identified operations.

X. Claims 31-34 and 38 depend either directly or indirectly from claim 30 and are allowable as depending from an allowable base claim, as well as for its own recited features.

Appellant states each of the above claims depending from claim 30 are allowable as depending from an allowable base claim. The rejections of these claims are maintained in light of the comments above in section **ix** with respect to claim 30.

(11) Related Proceeding(s) Appendix

No decision rendered by a court or the Board is identified by the examiner in the Related Appeals and Interferences section of this examiner's answer.

For the above reasons, it is believed that the rejections should be sustained.

Respectfully submitted,

/Rachna S Desai/
Primary Examiner, Art Unit 2176

Conferees:

Doug Hutton

/Doug Hutton/
Doug Hutton
Supervisory Primary Examiner
Technology Center 2100

Stephen Hong

/Stephen S. Hong/

Supervisory Patent Examiner, Art Unit 2178